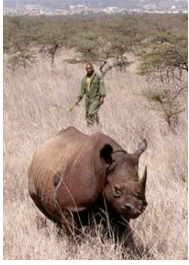


CS100J Fall 2007 Assignment A3 Monitoring Rhinos 2

Due (submitted on the CMS) by Sunday, 23 September



This assignment is a continuation of A1. Start with your A1 classes `Rhino` and `RhinoTester`, which should be correct. You can group all your test cases for this assignment in one test procedure.



You may work in groups of 2, but you must do all the work together, sitting at the computer together. For example, for one person to write functions and the other person to write test cases for them, with no interaction, is dishonest. If you want to form a group. *do it immediately*. Both partners must take action in the CMS before the group will be formed.

We set a date of Sunday, 23 September, so that this assignment will be completed before the first prelim. We know that you have a lot of work to do, including studying for the prelim. Therefore, we have reduced the work involved in this assignment by removing 4-5 functions from previous years (which had similar but not the same assignments).

In assignment A1, if you made a mistake, we asked you to fix it and resubmit. In this assignment, if you make a mistake, points will be deducted. And points may be deducted for lateness. Your new functions must have suitable javadoc comments on its methods, there must be appropriate test cases, and all methods should be correct. *You can achieve all this most easily by writing and testing one method at a time.*

In this assignment, you will add to the class of assignment 1: (1) A static variable, and (2) More comparison methods, using boolean expressions.

The steps to perform in doing this assignment

Step 1. Static variable. Add to class `Rhino` a **private static** variable that contains the number of `Rhino` objects that have been created so far. **WHENEVER A RHINO OBJECT IS CREATED, THIS FIELD SHOULD BE INCREASED BY 1.** And suitable test cases should be provided.

Add a **static int** function `getPopulation`, with no parameters, that will return the value of the static variable. Change the constructors so that they properly maintain the value of the static variable. Finally, add test cases to class `RhinoTester` to test whether the static variable is properly maintained.

Step 2. Comparison functions. Write the following functions, all of which produce a boolean value. **Do one at a time:** (1) write the function, then (2) test it thoroughly with test cases in class `RhinoTester`. **Then** proceed to the next function.

Method	Description
<code>isMother(Rhino r)</code>	= "r is not null, and r is this rhino's mother".
<code>isFather(Rhino r)</code>	= "r is not null, and r is this rhino's father".
<code>isParent(Rhino r)</code>	= "r is not null, and r is this rhino's parent".
<code>isBrotherOf(Rhino r)</code>	= "this rhino is the brother of r". Note: rhino A is called the brother of rhino B if the two are different, if A is male, and if they have at least one parent in common. Precondition: r is not null.
<code>isSisterOf(Rhino r)</code>	= "this rhino is the sister of r". Note: rhino A is called the sister of rhino B if the two are different, if A is female, and if they have at least one parent in common. Precondition: r is not null.

The names of your methods must match those listed above **exactly**, including capitalization. The number of parameters and their order must also match. The best way to ensure this is to copy and paste. Our testing will expect those method name and parameters, so any mismatch will fail during our testing. Parameter names will not be tested —you can change the parameter names if you want.

Each method **must** be preceded by an appropriate specification, as a Javadoc comment. The best way to ensure this is to copy and paste from this handout. After you have pasted, be sure to do any necessary editing.

Note carefully the definition of brother and sister in the specs. Use these definitions when writing the methods.

In testing whether two rhino's are the same, do not use the field that contains their names. Instead, use the names on the tabs of their manila folders.

Your new method bodies should have no if statements or conditional expressions.

Hints and directions

- Boolean expressions, using the operators `&&` (AND), `||` (OR), and `!` (NOT), are sufficient to implement all the functions. You will lose points for using `if` statements or conditional expressions.
- Some `Rhino` functions can be implemented easily using calls on previous ones. Take advantage of this as much as possible. (A few) points will be deducted if this is not done.
- Use method `.equals` to compare objects (including `String` objects) for equality and `==` to compare primitive values for equality.



Submitting the assignment

Check these points before you submit your assignment

- Did you use an `if` statement or conditional expression? Get rid of it.
- Is each method tested enough? For example, if an argument can be `null`, there should be at least one test case that has a call on the method with that argument being `null`; If not, points will be deducted.
- Did you check your javadoc? Click the javadoc button in the DrJava navigation bar. This will cause the specification of the classes and methods of the classes to be extracted from your program and html pages to be created that contain the specs. Look at those specs carefully and make sure that they are suitable. Can you understand precisely what a method does based on the extracted spec? If not, fix the spec, generate the javadoc, and look at it again.

Submit only files that end with the `.java`. Be careful about this, because in the same place as your `.java` files you may also have files that end with `.class` or `.java~`. but otherwise have the same name.